Below is a sample README file for your project:

---

# JourneyCompose

JourneyCompose is an Android application built using Jetpack Compose that enables users to plan a journey by selecting a starting city and a destination city. The application reads a list of stops from an asset file, calculates the route details, and provides visual cues—such as progress indication and visa requirements—to enhance the user experience.

## Features

- **Dynamic Journey Planning:**
  Users can enter a starting city and a destination city. The app then calculates the route based on stops loaded from an asset file.

- **Data Driven Stops:**
  The app reads stop information (city name, distance, visa requirement) from a file (`stops.txt`) located in the assets folder.

- **Adaptive UI:**
    - Uses a **LazyColumn** for routes with more than 3 stops to efficiently render the list.
    - Uses a scrollable **Column** for routes with 3 or fewer stops.
- **Visual Indicators:**
    - **Visa-Required Stops:** Displays a warning icon next to stops that require a visa.
    - **Progress Indication:**
        - Highlights stops that have been covered by fading them out.
        - The current stop is visually distinguished from upcoming stops.
    - **Progress Bar:** Displays the percentage of the journey completed and toggles between kilometers and miles.

## Project Structure

### 1. MainActivity.kt

- **Purpose:**
  The entry point of the app that sets up the Compose environment with edge-to-edge rendering.

- **Key Responsibilities:**

    - Loading the list of stops from `stops.txt` using the `loadStopsFromFile` function.
    - Initializing the navigation controller and passing the loaded stops to the navigation graph.

## 2. AppNavigation.kt

- **Purpose:**
  Defines the navigation graph for the app.

- **Key Responsibilities:**

    - Configuring navigation routes:
        - `startScreen`: For users to input the journey details.
        - `journeyScreen/{start}/{destination}`: To display the journey plan based on user input.

## 3. JourneyScreen.kt

- **Purpose:**
  Displays the journey details including progress and a list of stops.

- **Key Responsibilities:**

    - Validates the input cities against the list of stops.
    - Calculates route segments, total distance, and progress.
    - Renders the stops using a conditional layout:
        - **LazyColumn** for more than 3 stops.
        - **Scrollable Column** for 3 or fewer stops.
    - Provides visual cues for visa-required stops and already covered stops.
    - Offers controls to reset the journey or move to the next stop.

## 4. RouteSelection.kt (StartScreen)

- **Purpose:**
  Provides the user interface for inputting the starting and destination cities.

- **Key Responsibilities:**
    - Capturing user input via text fields.
    - Navigating to the `journeyScreen` with the entered data when the user taps the "Show Journey Plan" button.

### 5. Stop.kt

- **Purpose:**
  Defines the data model for a stop.

- **Key Responsibilities:**
    - Represents each stop with properties:
        - `name`: The name of the city.
        - `distanceKm`: The distance from the starting point.
        - `visaRequired`: A flag indicating if a visa is required.

### 6. stops.txt

- **Purpose:**
  Provides the data source for the stops.

**Format Example:**

New York,0,true
London,500,false
Paris,1300,true
Berlin,1750,false
Tokyo,2950,true
Sydney,4450,false
Dubai,5150,true
Singapore,5750,false

-

# Implementation Details

- **Data Loading:**
  The function `loadStopsFromFile` in `MainActivity.kt` reads the stops data from the `stops.txt` file, parses each line, and converts it into a list of `Stop` objects.

- **Navigation:**
  The app uses the Navigation Compose library to handle navigation between the start screen and the journey screen. User inputs are passed as arguments to the journey screen.

- **UI Rendering:**
  The journey screen adapts based on the number of stops:

    - **LazyColumn** is used for more than 3 stops to efficiently render long lists.
    - For 3 or fewer stops, a scrollable **Column** is used to avoid the overhead of a lazy list.
- **Visual Cues:**

    - Stops that require a visa have a warning icon next to their name.
    - Covered stops are faded (using an alpha modifier) to indicate progress.
    - The current stop is highlighted with a distinct background color.

# How to Run

1. **Clone the Repository:**
   Clone the repository to your local machine.

2. **Open in Android Studio:**
   Open the project in Android Studio.

3. **Build the Project:**
   Allow Android Studio to sync the project and build it.

4. **Run the App:**
   Deploy the app on an emulator or a physical Android device.

5. **Use the App:**

    - On the start screen, input the starting city and destination city.
    - Tap "Show Journey Plan" to view the journey details, including the list of stops and progress indicators.

# Conclusion

JourneyCompose is a modern Android application that leverages Jetpack Compose to provide an intuitive journey planning experience. Its dynamic UI, efficient data handling, and clear visual indicators make it an excellent starting point for more complex travel or route-planning applications.